



---

---


---

---

---

---

---

Tópicos Abordados 

- *WebClient*
- *WebRequest* e *WebResponse*
- *Uri* e *UriBuilder*
- Sockets
  - Tipos
    - TCP/IP
    - UDP/IP
  - Criando o servidor e o cliente
- *IPAddress*
- *Dns* e *IPHostEntry*

---

---


---

---

---

---

---

A Classe *WebClient* 

- A classe *System.Net.WebClient* é a forma mais fácil de acessar um arquivo localizado na internet
  - Não tem muitos recursos, devendo ser usada em casos simples

```
WebClient wc = new WebClient();  
wc.DownloadFile(  
    "http://www.softblue.com.br/public/images/sbv2_logotipo.png", "imagen.png");
```

Faz o download para um arquivo

```
Stream s = wc.OpenRead("http://www.softblue.com.br");
```

Abre uma stream de leitura

---

---

---

---

---

---

---

## As Classes *WebRequest* e *WebResponse*



- Executam requisições na internet através do uso de URIs
  - *WebRequest*: representa a requisição
  - *WebResponse*: representa a resposta da requisição
- Funcionam nativamente com alguns protocolos
  - HTTP(S)
  - FTP
  - FILE

---

---

---

---

---

---

---

## Fazendo uma Requisição



- Criação do objeto *WebRequest*

```
WebRequest req = WebRequest.Create("http://www.softblue.com.br");
```

O método *Create()* identifica o protocolo a ser usado

- Obtenção do objeto *WebResponse*

```
WebResponse resp = req.GetResponse();
```

---

---

---

---

---

---

---

## Mais Sobre o *WebRequest*



- Acesso com autenticação

```
req.Credentials = new NetworkCredential("username", "password");
```

- Presença de um servidor de proxy

```
req.Proxy = new WebProxy("192.168.1.1", true);
```

- Acesso a recursos do protocolo

```
HttpWebRequest httpReq = (HttpWebRequest) req;  
httpReq.UserAgent = "MyBrowser";
```

---

---


---

---

---

---

---

Mais Sobre o *WebResponse*


- Leitura dos headers

```

WebHeaderCollection headers = resp.Headers;
for (int i = 0; i < headers.Count; i++)
{
    string name = headers.GetKey(i);
    string[] values = headers.GetValues(i);
}

```
- Informações relativas ao protocolo

```

HttpWebResponse httpResp = (HttpWebResponse)resp;
HttpStatusCode code = httpResp.StatusCode;
string server = httpResp.Server;

```
- Leitura da resposta

```

Stream s = resp.GetResponseStream();

```

---

---

---


---

---

---

---

---

Requisições Assíncronas


- A requisição pode ser feita de forma assíncrona com o uso de delegate assíncrono

```

WebRequest req = WebRequest.Create("http://www.softblue.com.br");
req.BeginGetResponse(OnCompleted, req);

```

Delegate AsyncCallback

```

public void OnCompleted(IAsyncResult ar)
{
    WebRequest req = (WebRequest)ar.AsyncState;
    WebResponse resp = req.EndGetResponse(ar);
    ...
}

```

---

---

---


---

---

---

---

---

Criando URIs


- Uniform Resource Identifier
  - Todas as URLs são URIs
- Representada pela classe *Uri*

```

Uri uri = new Uri("http://www.softblue.com.br/site/comofunciona");

```
- Properties somente leitura que identificam os elementos da URI

| Property            | Resultado           |
|---------------------|---------------------|
| <i>Scheme</i>       | http                |
| <i>Host</i>         | www.softblue.com.br |
| <i>Port</i>         | 80                  |
| <i>AbsolutePath</i> | /site/comofunciona  |

---

---

---

---

---

---

---

---

## A Classe UriBuilder



- Permite criar uma URI através da especificação de suas partes

```
UriBuilder builder = new UriBuilder();  
builder.Scheme = "http";  
builder.Host = "www.softblue.com.br";  
builder.Port = 80;  
builder.Path = "/site/comofunciona";  
  
Uri uri = builder.Uri;
```

A classe UriBuilder tem outros construtores que podem facilitar a criação da URI

---

---

---

---

---

---

---

## Sockets



- Mecanismo de comunicação entre duas aplicações
- Baseado no modelo cliente/servidor
  - Uma aplicação servidor é executada numa determinada máquina e tem um socket ligado a uma porta específica dessa máquina
  - O servidor espera que um cliente faça um pedido de ligação através desse socket
- Dois tipos
  - TCP/IP
  - UDP/IP

---

---

---

---

---

---

---

## Sockets TCP/IP



- Existe uma conexão entre o cliente e o servidor
  - Permite envio de pacotes na forma de fluxo de dados (stream)
- A comunicação é confiável
  - Sem perda de dados
  - Sem inversão de ordem dos pacotes

---

---

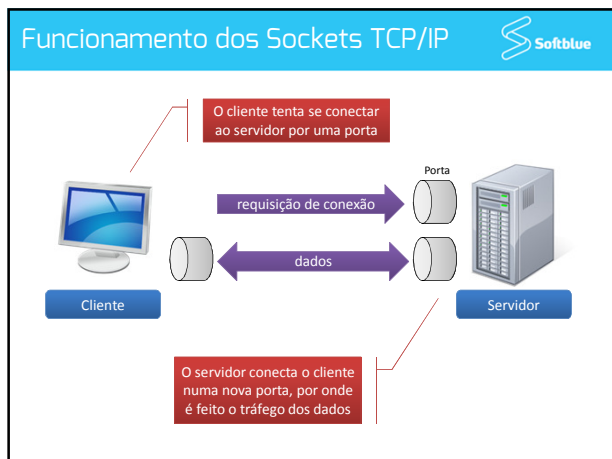
---

---

---

---

---




---

---

---

---

---

---

---

---

- ### Sockets UDP/IP
- Não existe uma conexão entre o cliente e o servidor
    - Envio de datagramas (remetente, receptor, conteúdo)
  - A comunicação não é confiável
    - Dados podem ser perdidos
    - Datagramas podem chegar fora de ordem
  - Muito mais veloz que sockets TCP/IP

---

---

---

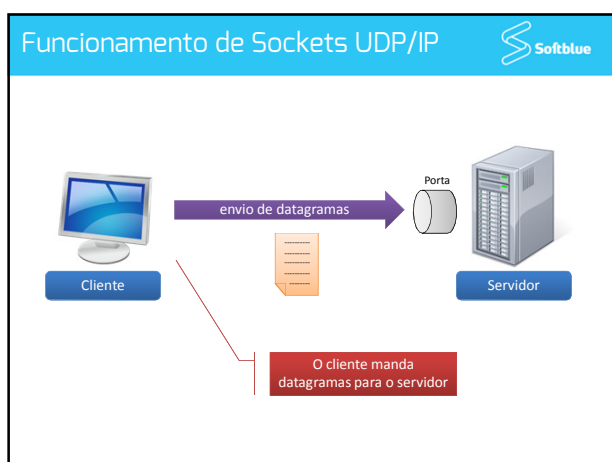
---

---

---

---

---




---

---

---

---

---

---

---

---

## Criando um Servidor TCP/IP

Softblue

- A classe *TcpListener* é usada
  - Namespace *System.Net.Sockets*

```

TcpListener server = new TcpListener(IPAddress.Loopback, 4000);
server.Start();

TcpClient client = server.AcceptTcpClient();
NetworkStream ns = client.GetStream();
  
```

localhost, porta 4000

Bloqueia até a chegada de uma conexão

Stream para leitura e escrita de dados

---

---

---

---

---

---

---

---

## Criando um Cliente TCP/IP

Softblue

- A classe *TcpClient* é usada

```

TcpClient client = new TcpClient("localhost", 4000);
NetworkStream ns = client.GetStream();
  
```

Estabelece a conexão usando um servidor e uma porta

Stream para leitura e escrita de dados

Uma exceção *SocketException* será lançada se a conexão não puder ser realizada

---

---

---

---

---

---

---

---

## Criando um Servidor UDP/IP

Softblue

- A classe *UdpClient* é usada

```

UdpClient server = new UdpClient(5000);
IPEndPoint remoteIP = new IPEndPoint(0, 0);
byte[] bytes = server.Receive(ref remoteIP);
string str = Encoding.ASCII.GetString(bytes, 0, bytes.Length);
  
```

Porta 5000

O método *Receive()* armazena os dados em *IPEndPoint*

O datagrama vem em forma de bytes

---

---

---


---

---

---

---

---

Criando um Cliente UDP/IP


- A classe *UdpClient* é usada

```

UdpClient client = new UdpClient();
string str = "Mensagem!";
byte[] bytes = Encoding.ASCII.GetBytes(str);
client.Send(bytes, bytes.Length, "localhost", 5000);

```

Converte para bytes

Envia para localhost na porta 5000

---

---

---


---

---

---

---

---

A Classe *IPAddress*


- Representa um endereço IP

```

IPAddress addr = IPAddress.Parse("192.168.1.1");

```

Cria o objeto com base no IP em forma de texto

```

IPAddress addr = IPAddress.Loopback;

```

Endereço de loopback (127.0.0.1)

```

IPAddress addr = IPAddress.Broadcast;

```

Endereço de broadcast (255.255.255.255)

---

---

---


---

---

---

---

---

As Classes *Dns* e *IPHostEntry*


- A classe *Dns* se comunica com o servidor de DNS configurado na rede

```

IPHostEntry he = Dns.GetHostEntry("www.softblue.com.br");
IPHostEntry he = Dns.GetHostEntry("177.71.183.253");

```

- O objeto *IPHostEntry* possui properties que fornecem informações sobre um host

| Property           | Descrição                           |
|--------------------|-------------------------------------|
| <i>HostName</i>    | Nome do host                        |
| <i>AddressList</i> | Lista de IPs associados ao host     |
| <i>Aliases</i>     | Lista de aliases associados ao host |

---

---

---

---

---

---

---

---



---

---

---

---

---

---

---